IN THE CLAIMS:

The claims have not been amended. The pending claims are reproduced below for the sake of convenience.

1. (Currently Amended) A computer implemented method including

scheduling tasks from a set thereof for running on a plurality of processors, each processor having access to a shared resource, wherein each task of the set of tasks is associated with one of a plurality of scheduling domains, at least one scheduling domain being associated with at least two tasks of the set of tasks, and wherein tasks within each scheduling domain can be run on different processors but are prohibited from running concurrently even if run on different processors; and

allowing a plurality of tasks of the set of tasks to run concurrently in different scheduling domains.

2. (Previously Presented) A method as in claim 1, including changing association for at least one task from a first to a second scheduling domain.

3. (Original) A method as in claim 1, including selecting for running at least one task associated with a plurality of said scheduling domains.

4. (Original) A method as in claim 1, including selecting for running at least one task not associated with any one of said scheduling domains.

5. (Currently Amended) A method including

altering a program code base, said program code base defining a plurality of tasks and a set of data structures at least some of which are shared, to include implicit synchronization among said tasks to said data structures, said implicit synchronization dividing said tasks into scheduling domains, wherein tasks within each scheduling domain can be run on different processors but are prohibited from running concurrently even if run on different processors, and

allowing multiple tasks to run concurrently.

6. (Currently Amended) A method including, in response to a program code base defining a plurality of tasks and a set of data structures at least some of which are shared,

altering said program code base to include program code or data associating each one of said tasks with one of a plurality of scheduling domains; and

providing a scheduler that (i) permits tasks within each scheduling domain to run on different processors but prohibits more than one task associated with the same scheduling domain from running concurrently even if run on different processors, and (ii) allows at least two tasks of the plurality of tasks to run concurrently, wherein each task of the at least two tasks is associated with a different scheduling domain.

7. (Previously Presented) A method as in claim 6, including altering said program code base to include instructions in at least one task changing association of said at least one task from a first to a second scheduling domain.

8. (Original) A method as in claim 6, including altering said program code base to include program code or data in at least one task associating said at least one task with a plurality of said scheduling domains.

9. (Original) A method as in claim 6, including altering said program code base to include program code or data in at least one task associating said at least one task with not any one of said scheduling domains.

10. (Original) A method as in claim 6, wherein said scheduler includes a plurality of runnable queues, one per scheduling domain.

11. (Currently Amended) A method including

running a plurality of tasks in a multiprocessor system;

implicitly synchronizing those tasks with regard to shared resources in said system by dividing said tasks into scheduling domains, wherein tasks within each scheduling domain can be run on different processors but are prohibited from running concurrently even if run on different processors; and

allowing multiple tasks of the plurality of tasks to run concurrently.

12. (Currently Amended) A system including

a plurality of processors, each processor of the plurality of processors having access to a shared resource;

a set of tasks, each task of the set of tasks being runnable on more than one of said processors, each said task being associated with one of a plurality of scheduling domains; and

each said processor including a scheduler that permits tasks within each scheduling domain to run on different processors but prohibits more than one task associated with the same scheduling domain from running concurrently even if run on different processors, while allowing a plurality of tasks of the set of tasks to run concurrently in different scheduling domains.

13. (Previously Presented) A system as in claim 12, having at least one task including instructions to change association of said at least one task from a first to a second scheduling domain.

14. (Original) A system as in claim 12, having at least one task runnable on more than one of said processors and associated with a plurality of said scheduling domains.

15. (Original) A system as in claim 12, having at least one task runnable on more than one of said processors and not associated with any one of said scheduling domains.

16. (Original) A system as in claim 12, wherein said scheduler includes a plurality of runnable queues, one per scheduling domain.

17. (Currently Amended) A system including a plurality of processors and a memory accessible to each processor of the plurality of processors, the memory storing a program code base, said program code base defining a plurality of tasks and a set of data structures at least some of which are shared, said program code base including implicit synchronization among said tasks to said data structures and concurrent execution of multiple tasks of the plurality of tasks, said implicit synchronization dividing said tasks into scheduling domains, wherein tasks within each scheduling domain can be run on different processors but are prohibited from running concurrently even if run on different processors.

18. (Currently Amended) A system including

a plurality of processors and a memory accessible to each processor of the plurality of processors, the memory storing code including code for

altering a program code base, said program code base defining a plurality of tasks and a set of data structures at least some of which are shared, to include program code or data associating each one of said tasks with one of a plurality of scheduling domains; and

providing a scheduler that (i) permits tasks within each scheduling domain to run on different processors but prohibits more than one task associated with the same scheduling domain from

running concurrently even if run on different processors, and (ii) allows at least two tasks of the plurality of tasks to run concurrently in different scheduling domains.

19. (Previously Presented) A system as in claim 18, said code including code for altering said program code base to include instructions in at least one task changing association of said at least one task from a first to a second scheduling domain.

20. (Previously Presented) A system as in claim 18, said code including code for altering said program code base to include program code or data in at least one task associating said at least one task with a plurality of said scheduling domains.

21. (Previously Presented) A system as in claim 18, said code including code for altering said program code base to include program code or data in at least one task associating said at least one task with not any one of said scheduling domains.

22. (Original) A system as in claim 18, wherein said scheduler includes a plurality of runnable queues, one per scheduling domain.

23. (Currently Amended) A process comprising

performing implicit synchronization of a plurality of tasks in a multiprocessor system, said implicit synchronization dividing said tasks into scheduling domains, wherein tasks within each

scheduling domain can be run on different processors but are prohibited from running concurrently even if run on different processors, and

allowing concurrent execution of multiple tasks of the plurality of tasks.

24. (Currently Amended) Memory or mass storage including

instructions in a set of tasks each runnable on more than one of a plurality of processors, each processor of the plurality of processors having access to a shared resource;

program code or data associating each of said tasks with one of a plurality of scheduling domains; and

scheduler instructions permitting tasks within each scheduling domain to run on different processors but preventing more than one task from running concurrently in the same scheduling domain even if run on different processors, said instructions allowing a plurality of tasks of the set of tasks to run concurrently in different scheduling domains.

25. (Canceled).

26. (Original) Memory or mass storage as in claim 24, including program code or data in at least one task associating said at least one task with a plurality of said scheduling domains.

27. (Original) Memory or mass storage as in claim 24, including program code or data in at least one task associating said at least one task with not any one of said scheduling domains.

28. (Previously Presented) Memory or mass storage as in claim 24, including instructions in at least one task changing association of said at least one task from a first to a second scheduling domain.

29. (Previously Presented) Memory or mass storage as in claim 24, wherein said scheduler instructions include instructions creating a plurality of runnable queues, one per scheduling domain.